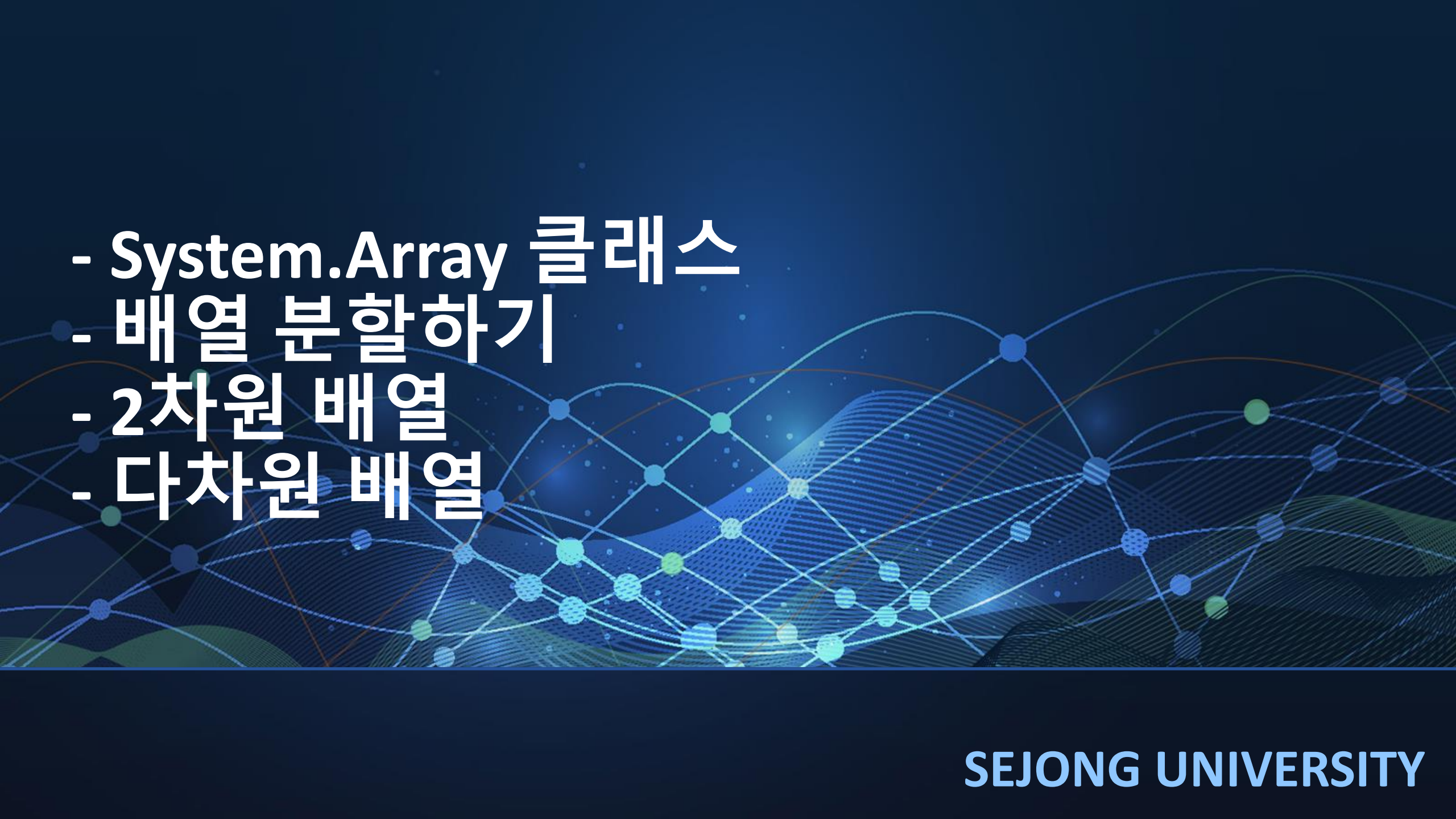


C# 프로그래밍

- 7주차 (2강)

- 
- System.Array 클래스
 - 배열 분할하기
 - 2차원 배열
 - 다차원 배열

System.Array 클래스

- .NET의 CTS (Common Type System)에서 배열은 System.Array 클래스에 대응
 - ✓ 아래 예제는 int 기반의 배열이 System.Array 형식에서 파생되었음을 보여줌

```
static void Main(string[] args)
{
    int[] array = new int[] { 10, 30, 20, 7, 1 };
    Console.WriteLine($"Type of array: {array.GetType()}");
    Console.WriteLine($"Base type of array: {array.GetType().BaseType}");
}
```

출력 결과:

Type of array: System.Int32[]
Base type of array: System.Array



System.Array 의 메소드와 프로퍼티

분류	이름	설명
정적 메소드	Sort ()	배열의 정렬
	BinarySearch<T> ()	이진 탐색을 수행
	IndexOf ()	특정 데이터의 인덱스를 반환
	TrueForAll ()	배열의 모든 요소가 지정한 조건 부합하는지 여부 반환
	FindIndex<T> ()	배열에서 지정한 조건에 부합하는 첫 번째 요소의 인덱스 반환
	Resize<T> ()	배열의 크기를 재조정
	Clear ()	배열의 모든 요소를 초기화
	ForEach<T> ()	배열의 모든 요소에 대해 동일한 작업을 수행
	Copy<T> ()	배열의 일부를 다른 배열에 복사
인스턴스 메소드	GetLength ()	배열에서 지정한 차원의 길이를 반환
프로퍼티	Length	배열의 길이를 반환
	Rank	배열의 차원을 반환



System.Array 의 메소드와 프로퍼티

분류	이름	설명
정적 메소드	Sort ()	배열의 정렬
	ForEach<T> ()	배열의 모든 요소에 대해 동일한 작업을 수행

```
private static void Print(int value)
{
    Console.WriteLine($"{value} ");
}

static void Main(string[] args)
{
    int[] scores = new int[] { 80, 74, 81, 90, 34 };

    foreach (int score in scores)
        Console.WriteLine($"{score} ");
    Console.WriteLine();

    Array.Sort(scores);
    Array.ForEach<int>(scores, new Action<int>(Print));
    Console.WriteLine();
}
```

출력 결과:

```
80 74 81 90 34
34 74 80 81 90
```



System.Array 의 메소드와 프로퍼티

분류	이름	설명
정적 메소드	Sort ()	배열의 정렬
	BinarySearch<T> ()	이진 탐색을 수행
	IndexOf ()	특정 데이터의 인덱스를 반환
프로퍼티	Rank	배열의 차원을 반환

```
static void Main(string[] args)
{
    int[] scores = new int[] { 80, 74, 81, 90, 34 };
    Array.Sort(scores);

    Console.WriteLine($"Number of dimensions: {scores.Rank}");

    Console.WriteLine($"Binary Search: 81 is at "
        + $"{Array.BinarySearch<int>(scores, 81)}");
    Console.WriteLine($"Binary Search: 90 is at "
        + $"{Array.IndexOf(scores, 90)}");
}
```

출력 결과:

```
Number of dimensions: 1
Binary Search: 81 is at 3
Binary Search: 90 is at 4
```



System.Array 의 메소드와 프로퍼티

분류	이름	설명
정적 메소드	Sort ()	배열의 정렬
	TrueForAll ()	배열의 모든 요소가 지정한 조건 부합하는지 여부 반환

```
private static bool CheckPassed(int score)
{
    return score >= 60;
}
```

```
static void Main(string[] args)
{
    int[] scores = new int[] { 80, 74, 81, 90, 34 };
    Array.Sort(scores);
```

```
// TrueForAll 메소드는 배열과 함께 조건을 검사하는 메소드를 매개변수로 받음
Console.WriteLine($"Everyone passed ? : "
    + $"{Array.TrueForAll<int>(scores, new Predicate<int>(CheckPassed))}");
}
```

출력 결과:

Everyone passed ? : False



System.Array 의 메소드와 프로퍼티

분류	이름	설명
정적 메소드	Sort ()	배열의 정렬
	TrueForAll ()	배열의 모든 요소가 지정한 조건 부합하는지 여부 반환
	FindIndex<T> ()	배열에서 지정한 조건에 부합하는 첫 번째 요소의 인덱스 반환

```
private static bool CheckPassed(int score)
{
    return score >= 60;
}
```

```
static void Main(string[] args)
{
    int[] scores = new int[] { 80, 74, 81, 90, 34 };
    Array.Sort(scores);
```

출력 결과:

Everyone passed ? : True

```
// FindIndex 메소드는 특정 조건에 부합하는 메소드를 매개변수로 받음
// 메소드는 람다식으로 구현
```

```
int index = Array.FindIndex<int>(scores, new Predicate<int>((score) => score < 60));
```

```
scores[index] = 61;
```

```
Console.WriteLine($"Everyone passed ? : "
    + $"{Array.TrueForAll<int>(scores, CheckPassed)}");
```

```
}
```



System.Array 의 메소드와 프로퍼티

분류	이름	설명
정적 메소드	Resize<T> ()	배열의 크기를 재조정
	ForEach<T> ()	배열의 모든 요소에 대해 동일한 작업을 수행
	Copy<T> ()	배열의 일부를 다른 배열에 복사
인스턴스 메소드	GetLength ()	배열에서 지정한 차원의 길이를 반환
프로퍼티	Length	배열의 길이를 반환

```
static void Main(string[] args)
{
```

```
    // ...
```

```
    Console.WriteLine("Old length of scores : "
        + $"{scores.GetLength(0)}");
```

```
    // 5 였던 배열 용량을 10으로 재조정
```

```
    Array.Resize<int>(ref scores, 10);
```

```
    Console.WriteLine($"New length of scores: {scores.Length}");
```

```
    Array.ForEach<int>(scores, new Action<int>(Print));
```

```
    Console.WriteLine();
```

```
    int[] sliced = new int[3];
```

```
    // scores 배열의 0번째부터 3개 요소를 sliced 배열의 0번째~2번째 요소로 복사
```

```
    Array.Copy(scores, 0, sliced, 0, 3);
```

```
    Array.ForEach<int>(sliced, new Action<int>(Print));
```

출력 결과:

Old length of scores : 5

New length of scores: 10

61 74 80 81 90 0 0 0 0 0

61 74 80



배열 분할하기

- C# 8.0에서 System.Range 형식 도입하여 범위를 나타냄
 - ✓ 시작 인덱스와 마지막 인덱스를 이용해서 범위를 나타냄
 - ✓ ".." 연산자 왼쪽에는 시작 인덱스, 오른쪽에는 마지막 인덱스 위치

시작 인덱스 ← → 마지막 인덱스

```
int[] scores = new int[] { 80, 74, 81, 90, 34 };  
System.Range r1 = 0..3;  
int[] sliced = scores[r1]; // [와 ] 사이에 인덱스 대신 System.Range 객체를 입력하여 분할된 배열 반환  
  
int[] sliced2 = scores[0..3]; // [와 ] 사이에 직접 .. 연산자를 입력하면 코드가 간결해짐
```



배열 분할하기

- .. 연산자의 오른쪽 마지막 인덱스는 분할 결과에 제외
 - ✓ 왼쪽 시작 인덱스는 분할 결과에 포함
- .. 연산자의 두 피연산자는 생략 가능
 - ✓ 시작 인덱스를 생략하면 배열의 첫 번째 요소의 위치를 시작 인덱스로 간주
 - ✓ 마지막 인덱스를 생략하면 마지막 요소의 위치를 마지막 인덱스로 간주
 - ✓ 시작과 마지막 인덱스를 모두 생략하면 배열 전체를 나타냄

```
// 첫 번째(0)부터 세 번째(2) 요소까지
int[] sliced3 = scores[..3];

// 첫 번째(1)부터 마지막 요소까지
int[] sliced4 = scores[1..];

// 전체
int[] sliced5 = scores[..];
```

System.Index 객체 이용

```
System.Index idx = ^1;
int[] sliced6 = scores[..idx];

int[] sliced7 = scores[..^1];
```



배열 분할 예제코드

```
static void Main(string[] args)
{
    char[] array = new char['Z' - 'A' + 1]; // 대문자 알파벳 개수
    // array에 'A'부터 'Z'까지 입력
    for (int i = 0; i < array.Length; i++)
        array[i] = (char)('A' + i);

    PrintArray(array[..]); // 0부터 마지막까지
    PrintArray(array[5..]); // 5부터 끝까지

    Range range_5_10 = 5..10;
    PrintArray(array[range_5_10]); // 5부터 9번째까지

    Index last = ^0;
    Range range_5_last = 5..last;
    PrintArray(array[range_5_last]); // 5부터 끝까지

    PrintArray(array[^4..^1]); // 끝에서 4번째부터 끝에서 2번째까지
}
```

```
static void PrintArray(System.Array array)
{
    foreach (var e in array)
        Console.Write(e);
    Console.WriteLine();
}
```

출력 결과:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
FGHIJKLMNOPQRSTUVWXYZ
FGHIJ
FGHIJKLMNOPQRSTUVWXYZ
WXY
```

2차원 배열 선언

- 2개의 차원(세로+가로)으로 원소를 배치하는 2차원 배열
 - ✓ 1차원 배열을 원소로 갖는 배열

사용형식

```
데이터형식[ , ] 배열이름 = new 데이터형식[2차원길이, 1차원길이];
```

선언 예제

```
int [,] arr = new int[3,10];
```

arr[0,0]	arr[0,1]	arr[0,2]	arr[0,3]	arr[0,4]	arr[0,5]	arr[0,6]	arr[0,7]	arr[0,8]	arr[0,9]
arr[1,0]	arr[1,1]	arr[1,2]	arr[1,3]	arr[1,4]	arr[1,5]	arr[1,6]	arr[1,7]	arr[1,8]	arr[1,9]
arr[2,0]	arr[2,1]	arr[2,2]	arr[2,3]	arr[2,4]	arr[2,5]	arr[2,6]	arr[2,7]	arr[2,8]	arr[2,9]



2차원 배열 초기화

- 2차원 배열의 원소 접근
 - ✓ 첫 번째 차원과 두 번째 차원의 인덱스를 대괄호 [와] 사이에 같이 입력

2차원 배열 선언 및 각 원소에 값 할당 예제

```
int[,] array = new int[2, 3];  
array[0, 0] = 1;  
array[0, 1] = 2;  
array[0, 2] = 3;  
array[1, 0] = 4;  
array[1, 1] = 5;  
array[1, 2] = 6;
```

2차원 배열의 세 가지 초기화 방법

```
int[,] arr = new int[2, 3] { { 1, 2, 3 }, { 4, 5, 6 } }; // 배열의 형식과 길이 명시  
int[,] arr2 = new int[,] { { 1, 2, 3 }, { 4, 5, 6 } }; // 배열의 길이를 생략  
int[,] arr3 = { { 1, 2, 3 }, { 4, 5, 6 } }; // 형식과 길이를 모두 생략
```



2차원 배열 예제코드

2차원 배열의 첫 번째 초기화 방법

```
static void Main(string[] args)
{
    int[,] arr = new int[2, 3] { { 1, 2, 3 }, { 4, 5, 6 } }; // 배열의 형식과 길이 명시
    for (int i = 0; i < arr.GetLength(0); i++)
    {
        for (int j = 0; j < arr.GetLength(1); j++)
        {
            Console.Write($"[{i}, {j}] : {arr[i, j]} ");
        }
        Console.WriteLine();
    }
    Console.WriteLine();
}
```

출력 결과:

```
[0, 0] : 1 [0, 1] : 2 [0, 2] : 3
[1, 0] : 4 [1, 1] : 5 [1, 2] : 6
```



2차원 배열 예제코드

2차원 배열의 두 번째 초기화 방법

```
static void Main(string[] args)
{
    int[,] arr2 = new int[,] { { 1, 2, 3 }, { 4, 5, 6 } }; // 배열의 길이를 생략
    for (int i = 0; i < arr2.GetLength(0); i++)
    {
        for (int j = 0; j < arr2.GetLength(1); j++)
        {
            Console.WriteLine($"{i}, {j} : {arr2[i, j]} ");
        }
        Console.WriteLine();
    }
    Console.WriteLine();
}
```

출력 결과:

```
[0, 0] : 1 [0, 1] : 2 [0, 2] : 3
[1, 0] : 4 [1, 1] : 5 [1, 2] : 6
```



2차원 배열 예제코드

2차원 배열의 세 번째 초기화 방법

```
static void Main(string[] args)
{
    int[,] arr3 = { { 1, 2, 3 }, { 4, 5, 6 } }; // 형식과 길이를 모두 생략
    for (int i = 0; i < arr3.GetLength(0); i++)
    {
        for (int j = 0; j < arr3.GetLength(1); j++)
        {
            Console.WriteLine($"[{i}, {j}] : {arr3[i, j]} ");
        }
        Console.WriteLine();
    }
    Console.WriteLine();
}
```

출력 결과:

```
[0, 0] : 1 [0, 1] : 2 [0, 2] : 3
[1, 0] : 4 [1, 1] : 5 [1, 2] : 6
```



다차원 배열 선언

- 차원이 둘 이상인 배열
 - ✓ 사용하는 인덱스 수가 2개, 3개, 4개, .. 로 증가

사용형식

데이터형식[, ,] 배열이름 = new 데이터형식[3차원길이, 2차원길이, 1차원길이];

선언 예제

```
int [, , ] arr = new int[4,3,10];
```

arr[0,0,0]	arr[0,0,1]	arr[0,0,2]	arr[0,0,3]	arr[0,0,4]	arr[0,0,5]	arr[0,0,6]	arr[0,0,7]	arr[0,0,8]	arr[0,0,9]
arr[1,0,0]	arr[1,0,1]	arr[1,0,2]	arr[1,0,3]	arr[1,0,4]	arr[1,0,5]	arr[1,0,6]	arr[1,0,7]	arr[1,0,8]	arr[1,0,9]
arr[2,0,0]	arr[2,0,1]	arr[2,0,2]	arr[2,0,3]	arr[2,0,4]	arr[2,0,5]	arr[2,0,6]	arr[2,0,7]	arr[2,0,8]	arr[2,0,9]
arr[3,0,0]	arr[3,0,1]	arr[3,0,2]	arr[3,0,3]	arr[3,0,4]	arr[3,0,5]	arr[3,0,6]	arr[3,0,7]	arr[3,0,8]	arr[3,0,9]
arr[3,1,0]	arr[3,1,1]	arr[3,1,2]	arr[3,1,3]	arr[3,1,4]	arr[3,1,5]	arr[3,1,6]	arr[3,1,7]	arr[3,1,8]	arr[3,1,9]
arr[3,2,0]	arr[3,2,1]	arr[3,2,2]	arr[3,2,3]	arr[3,2,4]	arr[3,2,5]	arr[3,2,6]	arr[3,2,7]	arr[3,2,8]	arr[3,2,9]



다차원 배열 예제코드

```
int[, ,] arr = new int[4, 3, 2] // 배열의 형식과 길이 명시
{
    { {1,2}, {3, 4 }, { 5, 6 } },
    { { 1,4}, { 2, 5 }, { 3, 6 } },
    { { 6,5}, { 4, 3 }, { 2, 1 } },
    { { 6,3}, { 5, 2 }, { 4, 1 } },
};

for (int i = 0; i < arr.GetLength(0); i++)
{
    for (int j = 0; j < arr.GetLength(1); j++)
    {
        Console.Write("{ ");
        for (int k = 0; k < arr.GetLength(2); k++)
            Console.Write($"{arr[i, j, k]} ");
        Console.Write("} ");
    }
    Console.WriteLine();
}
```

출력 결과:

{ 1 2 }	{ 3 4 }	{ 5 6 }
{ 1 4 }	{ 2 5 }	{ 3 6 }
{ 6 5 }	{ 4 3 }	{ 2 1 }
{ 6 3 }	{ 5 2 }	{ 4 1 }





Thank you!